

Towards a Calculus of Echo State Networks

Alireza Goudarzi¹ and Darko Stefanovic^{1,2}

¹ Department of Computer Science, University of New Mexico, Albuquerque, New Mexico, USA
alirezag@cs.unm.edu

² Center for Biomedical Engineering, University of New Mexico, Albuquerque, New Mexico, USA
darko@cs.unm.edu

Abstract

Reservoir computing is a recent trend in neural networks which uses the dynamical perturbations on the phase space of a system to compute a desired target function. We present how one can formulate an expectation of system performance in a simple class of reservoir computing called echo state networks. In contrast with previous theoretical frameworks, which only reveal an upper bound on the total memory in the system, we analytically calculate the entire memory curve as a function of the structure of the system and the properties of the input and the target function. We demonstrate the precision of our framework by validating its result for a wide range of system sizes and spectral radii. Our analytical calculation agrees with numerical simulations. To the best of our knowledge this work presents the first exact analytical characterization of the memory curve in echo state networks.

Keywords: Reservoir computing, echo state networks, analytical training, Wiener filters, dynamics

1 Introduction

In this paper we report our preliminary results in building a framework for a mathematical study of reservoir computing (RC) architecture called the echo state network (ESN). Reservoir computing (RC) is a recent approach in time series analysis that uses the perturbations in the intrinsic dynamics of a system, as opposed to its stable states, to compute a desired function. The classic example of reservoir computing is the echo state network, a recurrent neural network with random structure. These networks have shown good performance in many signal processing applications. The theory of echo state networks consists of analysis of memory capacity [4], e.g., how long can the network remember its inputs, and the echo state property [18], which consist of analysis of long term convergence of the phase space of the network. In RC, computation relies on the dynamics of the system and not its specific structure, which makes the approach an intriguing paradigm for computing with unconventional and neuromorphic architectures [13–15]. In this context, our vision is to develop special-purpose computing devices that can be trained or “programmed” to perform a specific task. Consequently, we would like to know the expected performance of a device with a given structure

on given a task. Echo state networks (ESN) give us a simple model to study reservoir computing. Extant studies of computational capacity and performance of ESN for various tasks have been carried out computationally and the main theoretical insight has been the upper bound for linear memory capacity [5, 12, 17].

Our aim is to use ESN to develop a theoretical framework that allows us to form an expectation about the performance of RC for a desired computation. To demonstrate the power of this framework, we apply it to the problem of memory curve characterization in ESN. Whereas previous attempts used the annealed approximation method to simplify the problem [17], we derive an exact analytical solution to characterize the memory curve in the system. Our formulation reveals that ESN computes an output as a linear combination of the correlation structure of the input signal and therefore the performance of ESN on a given task will depend on how well the output can be described as the input correlation in various time scales. Full development of the framework will allow us to extend our predictions to more complex tasks and more general RC architectures.

2 Background

In RC, a high-dimensional dynamical core called a *reservoir* is perturbed with an external input. The reservoir states are then linearly combined to create the output. The readout parameters can be calculated by performing regression on the state of a teacher-driven reservoir and the expected teacher output. Figure 1 shows an RC architecture. Unlike other forms of neural computation, computation in RC takes place within the transient dynamics of the reservoir. The computational power of the reservoir is attributed to a short-term memory created by the reservoir [9] and the ability to preserve the temporal information from distinct signals over time [10, 11]. Several studies attributed this property to the dynamical regime of the reservoir and showed it to be optimal when the system operates in the critical dynamical regime—a regime in which perturbations to the system’s trajectory in its phase space neither spread nor die out [1–3, 11, 16]. The reason for this observation remains unknown. Maass et al. [10] proved that given the two properties of *separation* and *approximation*, a reservoir system is capable of approximating any time series. The separation property ensures that the reservoir perturbations from distinct signals remain distinguishable, whereas the approximation property ensures that the output layer can approximate any function of the reservoir states to an arbitrary degree of accuracy. Jaeger [8] proposed that an ideal reservoir needs to have the so-called echo state property (ESP), which means that the reservoir states asymptotically depend on the input and not the initial state of the reservoir. It has also been suggested that the reservoir dynamics acts like a spatiotemporal kernel, projecting the input signal onto a high-dimensional feature space [6].

3 Model

In this paper we restrict attention to linear ESNs, in which both the transfer function of the reservoir nodes and the output layer are linear functions, Figure 2. The readout layer is usually a linear combination of the reservoir states. The readout weights are determined using supervised learning techniques, where the network is driven by a teacher input and its output is compared with a corresponding teacher output to estimate the error. Then, the weights can be calculated using any closed-form regression technique [12] in offline training contexts.

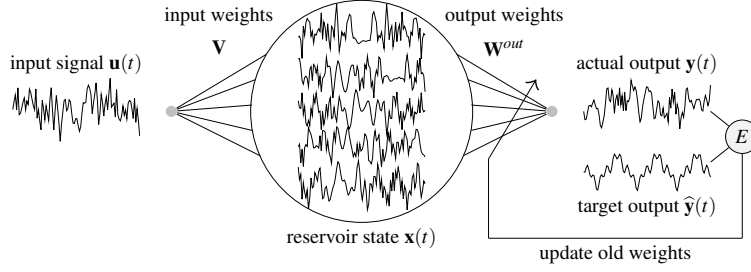


Figure 1: Computation in a reservoir computer (RC). The reservoir is an excitable dynamical system with N readable output states represented by the vector $\mathbf{x}(t)$. The input signal $\mathbf{u}(t)$ is fed into one or more points i in the reservoir with a corresponding weight w_i^{in} denoted with weight column vector $\mathbf{W}^{in} = [w_i^{in}]$.

Mathematically, the input-driven reservoir is defined as follows. Let N be the size of the reservoir. We represent the time-dependent inputs as a column vector $\mathbf{u}(t)$, the reservoir state as a column vector $\mathbf{x}(t)$, and the output as a column vector $\mathbf{y}(t)$. The input connectivity is represented by the matrix \mathbf{V} and the reservoir connectivity is represented by an $N \times N$ weight matrix \mathbf{W} . For simplicity, we assume one input signal and one output, but the notation can be extended to multiple inputs and outputs. The time evolution of the linear reservoir is given by:

$$\mathbf{x}(t+1) = \mathbf{W}\mathbf{x}(t) + \mathbf{V}\mathbf{u}(t), \quad (1)$$

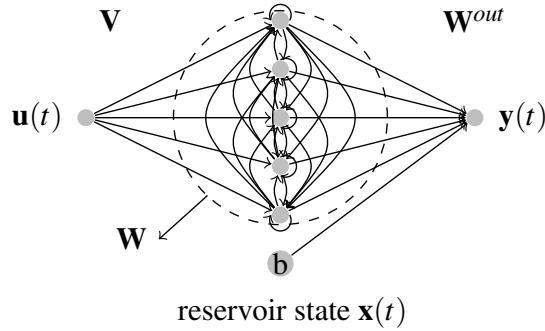


Figure 2: Schematic of an echo state network (ESN). A dynamical core called a reservoir is driven by input signal $\mathbf{u}(t)$. The states of the reservoir $\mathbf{x}(t)$ extended by a constant 1 and combined linearly to produce the output $\mathbf{y}(t)$. The reservoir consists of N nodes interconnected with a random weight matrix \mathbf{W} . The connectivity between the input and the reservoir nodes is represented with a randomly generated weight matrix \mathbf{W}^{in} . The reservoir states and the constant are connected to the readout layer using the weight matrix \mathbf{W}^{out} . The reservoir and the input weights are fixed after initialization, while the output weights are learned using a regression technique.

The output is generated by the multiplication of an output weight matrix \mathbf{W}^{out} of length N and the reservoir state vector $\mathbf{x}(t)$:

$$\mathbf{y}(t) = \mathbf{W}^{out} \mathbf{x}(t). \quad (2)$$

The coefficient vector \mathbf{W}^{out} is calculated to minimize the squared output error $E = \langle \|\mathbf{y}(t) - \hat{\mathbf{y}}(t)\|^2 \rangle$ given the target output $\hat{\mathbf{y}}(t)$. Here, $\|\cdot\|$ is the L_2 norm and $\langle \cdot \rangle$ the time average. The output weights are calculated using ordinary linear regression using a pseudo-inverse form:

$$\mathbf{W}^{out} = \langle \mathbf{X} \mathbf{X}' \rangle^{-1} \langle \mathbf{X} \hat{\mathbf{Y}}' \rangle, \quad (3)$$

where each row t in the matrix \mathbf{X} corresponds to the state vector $\mathbf{x}(t)$, and $\hat{\mathbf{Y}}$ is the target output matrix, whose rows correspond to target output vectors $\hat{\mathbf{y}}(t)$.

4 Deriving the Expected Performance

Our goal is to form an expectation for the performance based on the structure of the reservoir and the properties of the task. Our approach is to calculate an expected output weight \mathbf{W}^{out} using which the error of the system can be estimated. The calculation of \mathbf{W}^{out} using the regression equation Equation 3 has two components: the Gramm matrix $\langle \mathbf{X} \mathbf{X}' \rangle$, and the projection matrix $\mathbf{X}^T \hat{\mathbf{y}}$. We now show how each of the two can be computed.

To compute the Gramm matrix, we start by expanding the recursion in Equation 1 to get an explicit expression for $\mathbf{x}(t)$ in terms of the initial condition and the input history:

$$\mathbf{x}_t = \mathbf{W}^t \mathbf{x}_0 + \sum_{i=0}^{t-1} \mathbf{W}^{t-i-1} \mathbf{V} u_i, \quad (4)$$

$$(5)$$

Note that we have written the time as subscript for readability. The first term in this expression is the contributions from the initial condition of the reservoir which will vanish for large t when $\lambda^{max} < 1$, where λ^{max} is the spectral radius of the reservoir. Therefore without loss of generality we can write:

$$\mathbf{x}_t = \sum_{i=0}^{t-1} \mathbf{W}^{t-i-1} \mathbf{V} u_i, \quad (6)$$

$$(7)$$

Now we can expand the Gramm matrix as follows:

$$\langle \mathbf{X} \mathbf{X}' \rangle = \frac{1}{T} \sum_{t=0}^T \mathbf{x}_t \otimes \mathbf{x}'_t, \quad (8)$$

$$= \frac{1}{T} \sum_{t=0}^T \left(\sum_{i=0}^{t-1} \mathbf{W}^{t-i-1} \mathbf{V} u_i \right) \otimes \left(\sum_{j=0}^{t-1} \mathbf{W}^{t-j-1} \mathbf{V} u_j \right)' \quad (9)$$

$$= \frac{1}{T} \sum_{t=0}^T \left(\sum_{i=0}^{t-1} \sum_{j=0}^{t-1} \mathbf{W}^{t-i-1} \mathbf{V} u_i \otimes u'_j \mathbf{V}' \mathbf{W}^{t-j-1} \right) \quad (10)$$

The symbol \otimes is the product. Similarly, we can write the projection matrix as:

$$\langle \mathbf{X}\hat{\mathbf{Y}}' \rangle = \frac{1}{T} \sum_{t=0}^T \mathbf{x}_t \hat{y}_t' \quad (11)$$

$$= \frac{1}{T} \sum_{t=0}^T \left(\sum_{i=0}^{t-1} \mathbf{W}^{t-i-1} \mathbf{V} u_i \right) \otimes \hat{y}_t' \quad (12)$$

$$= \frac{1}{T} \sum_{t=0}^T \sum_{i=0}^{t-1} \mathbf{W}^{t-i-1} \mathbf{V} u_i \otimes \hat{y}_t' \quad (13)$$

Note that in Equation 10 and Equation 13, we have an explicit dependence on the reservoir weight matrix \mathbf{W} , the input weight matrix \mathbf{V} , the autocorrelation of input signal $u_i \otimes u_j$, and correlation of input and output $u_i \otimes \hat{y}_t'$, at various time scales. In the next section we use our fundamental equations, Equation 10 and 13, to calculate \mathbf{W}^{out} and ultimately the expected performance of the system on a given task.

5 Computing the Memory Curve

In this section, we use our derivation to analytically calculate the memory capacity curve for a given reservoir structure defined by \mathbf{W} and \mathbf{V} . Our assumption is that \mathbf{W} is non-singular and has spectral radius $\lambda^{max} < 1$. Memory capacity is the ability of the ESN to reconstruct its input after a delay of τ . It is defined as [7]:

$$MC_\tau = \frac{\text{Cov}^2(u_{t-\tau}, y_t)}{\text{Var}(u_t) \text{Var}(y_t)}, \quad (14)$$

where u_t is the input at time t , $u_{t-\tau}$ is the corresponding target output, and y_t is the output of the network after calculating the \mathbf{W}_{out} . The inputs u_t are drawn from identical and independent uniform distributions in the range $[-1, 1]$. The total memory capacity of a network is then given by:

$$MC = \sum_{\tau=1}^{\infty} MC_\tau. \quad (15)$$

We now demonstrate how we can use the structure of the memory function and the i.i.d. property to analytically compute each component of \mathbf{W}_{out} .

To compute $\langle \mathbf{X}\mathbf{X}' \rangle$, we can assume that we have access to a one-dimensional infinitely long input-output stream to calculate the output weights. Then we can write:

$$\langle \mathbf{X}\mathbf{X}' \rangle = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^T \left(\sum_{i=0}^{t-1} \sum_{j=0}^{t-1} \mathbf{W}^{t-i-1} \mathbf{V} u_i \otimes u_j \mathbf{V}' \mathbf{W}^{t-j-1} \right) \quad (16)$$

$$= \sum_{i=0}^{t-1} \sum_{j=0}^{t-1} \langle u_i u_j \rangle \mathbf{W}^{t-i-1} \mathbf{V} \otimes \mathbf{V}' \mathbf{W}^{t-j-1} \quad (17)$$

Since u_i are drawn from i.i.d. uniform distribution between $[-1, 1]$, we have:

$$\langle u_i u_j \rangle = \begin{cases} 0, & \text{if } i \neq j \\ \frac{1}{3}, & \text{if } i = j \end{cases} \quad (18)$$

Moreover, let $\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{U}^{-1}$ be the eigenvalue decomposition of \mathbf{W} , \mathbf{d} be a column vector corresponding to the diagonal elements of \mathbf{D} , $\tilde{\mathbf{V}} = \mathbf{U}^{-1}\mathbf{V}$ be the input weights in the basis defined by the eigenvectors of \mathbf{W} , and \mathbf{I}° the identity of Hadamard product denoted by \circ . We can rewrite Equation 17 as follows:

$$\langle \mathbf{X}\mathbf{X}' \rangle = \mathbf{U} \left(\lim_{t \rightarrow \infty} \sum_{i=0}^{t-1} \langle u_i u_i \rangle \tilde{\mathbf{V}} \otimes \tilde{\mathbf{V}}' \circ \mathbf{d}^{i-1} \otimes \mathbf{d}'^{i-1} \right) \mathbf{U}' \quad (19)$$

$$= \langle u^2 \rangle \mathbf{U} (\tilde{\mathbf{V}} \otimes \tilde{\mathbf{V}}' \circ \tilde{\mathbf{d}} \otimes \tilde{\mathbf{d}}') \mathbf{U}' \quad (20)$$

Here $\langle u^2 \rangle = \frac{1}{3}$ is the variance of the input, and $\tilde{\mathbf{d}}$ is a column vector whose elements are $\tilde{d}_l = \frac{1}{1-d_l}$. We have thus, computed the Gramm matrix as a function of the variance of the input, \mathbf{W} , and \mathbf{V} . Note that for MC_τ , $\langle \mathbf{X}\mathbf{X}' \rangle$ is constant for all τ . However, for each τ we have to calculate $\langle \mathbf{X}\hat{\mathbf{Y}}' \rangle_\tau$ separately, denoted by the subscript τ . Under the assumption of i.i.d. input, the calculation of $\langle \mathbf{X}\hat{\mathbf{Y}}' \rangle_\tau$ directly follows from Equation 13:

$$\langle \mathbf{X}\hat{\mathbf{Y}}' \rangle_\tau = \langle u^2 \rangle \mathbf{W}^{\tau-1} \mathbf{V}. \quad (21)$$

Our analytical solution of \mathbf{W}_{out} assumes the true variance of the input. By appeal to the central limit theorem, we can expect that if one calculates the memory curve for a given ESN numerically, due to finite training size, the results should vary according to a normal distribution around the analytical values. We should note that previous attempts to characterize the memory curve of ESN [17] used an annealed approximation over the Gaussian Orthogonal Ensemble (GOE) to simplify the problem. Here, in contrast, we presented an exact solution to this problem.

6 Results

In this section we calculate the memory capacity of a given ESN using our analytical solution and compare it with numerical estimations. For the purpose of demonstration, we create ten $N \times N$ reservoir weight matrices \mathbf{W} and corresponding $N \times 1$ input weight matrix \mathbf{V} by sampling a zero-mean normal distribution with standard deviation of 1. We then rescale the weight matrix to have a spectral radius of $\lambda^* = \lambda$. For each $\{\mathbf{W}, \mathbf{V}\}$ pair we run the system with an input stream of length 5000. We discard the first 2000 reservoir states and use the rest to calculate MC_τ , and repeat this experiment 10 times to calculate the average the τ -delay memory capacity \overline{MC}_τ . As we will see, the variance in our result is low enough for 10 runs to give us a reliable average behavior. We choose $1 \leq \tau \leq 100$, and try the experiment with $N \in \{25, 50, 75, 100\}$ and $\lambda \in \{0.1, 0.50, 0.95\}$.

Figure 3a and Figure 3b illustrate the probability distribution of our simulated results for the entries of $\langle \mathbf{X}\mathbf{X}' \rangle$ and $\langle \mathbf{X}\hat{\mathbf{Y}}' \rangle$ for a sample ESN with $N = 50$ nodes and spectral radius $\lambda = 0.95$. We drove the ESN with 20 different input time series and for each input time series we calculated the matrices $\langle \mathbf{X}\mathbf{X}' \rangle$, $\langle \mathbf{X}\hat{\mathbf{Y}}' \rangle$. To look at all the entries of $\langle \mathbf{X}\mathbf{X}' \rangle$ at the same time we create a dataset $\mathbf{X}\mathbf{X}'^*$ by shifting and rescaling each entry of $\langle \mathbf{X}\mathbf{X}' \rangle$ with the corresponding analytical values so all entries map onto a zero-mean normal distribution. As expected there is no skewness in the result, suggesting that all values follow a normal distribution centered at the analytical calculations for each entry. Similarly for Figure 3b, we create a dataset $\mathbf{X}\hat{\mathbf{Y}}'^*$ by shifting and rescaling each entry of $\langle \mathbf{X}\hat{\mathbf{Y}}' \rangle$ with the corresponding analytical values to observe that all values follow a normal distribution centered at the analytical values with no skewness.

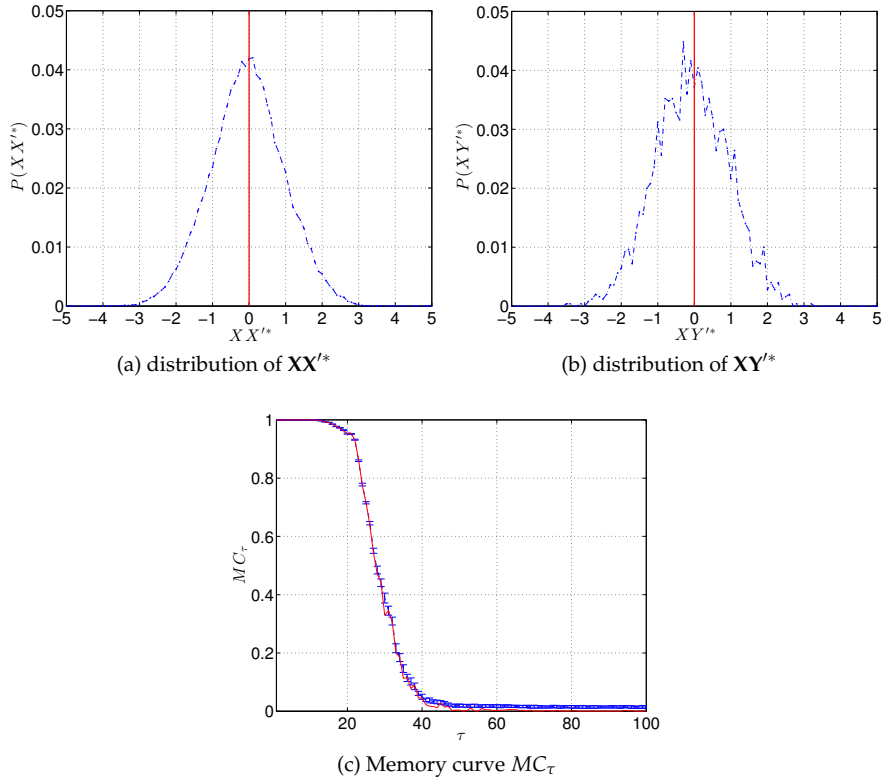


Figure 3: Agreement between analytical and simulation results for $\langle XX' \rangle$ (a), $\langle XY' \rangle$ (b) and the memory curve MC_τ (c).

Figure 3c shows the complete memory curve MC_τ for the sample ESN. Our analytical results (solid red line) are in good agreement with simulation results (solid blue line). Note that our results are exact calculations and not approximation, therefore the analytical MC_τ curve also replicates the fluctuations for various values of MC_τ that are *signatures* of a particular instantiation of the ESN model.

Next, we analyze the accuracy of our analytical results with respect to changes in the reservoir size N and its spectral radius λ . Figure 4 shows the result of this analysis, and reveals two interesting trends for accuracy and memory behavior for different N and λ . For all N and λ the analytical calculation of MC_τ agrees very well with the numerical simulation. However, as we approach $\lambda = 1$, the variance in the simulation result increases during the phase transition from $MC_\tau = 1$ to $MC_\tau = 0$, likely because the reservoir approaches the onset of chaos, e.g., $\lambda = 1$.

The behavior of the memory function also shows interesting behavior. For small $N < 50$, the transition from high to low MC_τ occurs very close to $\tau = N$, as expected from the fundamental limit $MC \leq N$. However, as the reservoir size grows, the position of the transition in MC_τ diverges from N . Note that our analytical calculation is equivalent to using infinite size training data for calculating the output weights, therefore divergence of the actual memory capacity from the bound N cannot be attributed to finite training size. Determining the reason for this discrepancy requires a more careful analysis of the memory function.

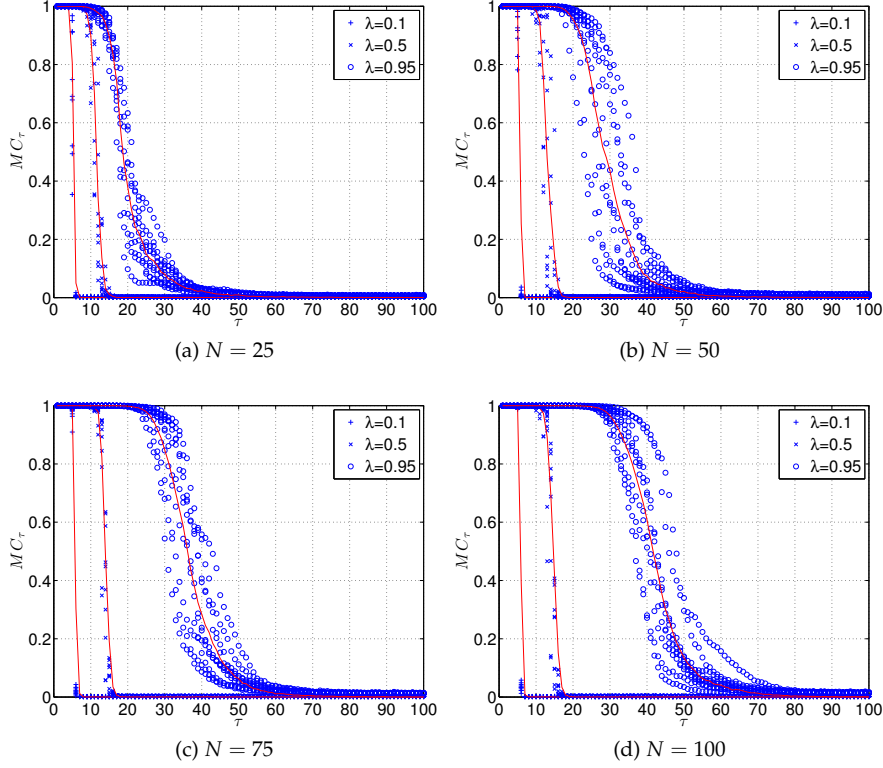


Figure 4: Sensitivity of the analytical (red solid lines) and simulation results (data markers) for the memory curve MC_τ to changes in the system size N and the spectral radius λ . The data were generated from 20 systems each driven with 20 different input streams. For all N and λ values, the analytical and simulated results are in good agreement. However, as the spectral radius approaches $\lambda = 1$ the variance of the simulated results increases, suggesting that the system is approaching the chaotic dynamical phase.

7 Conclusion and Future Work

Our aim is to go beyond only the memory bounds in dynamical system and develop a rigorous understanding of the expected memory of the system given its structure and a desired input domain. Here, we have built a formal framework that expresses the memory as a function of the system structure and autocorrelation structure of the input. Previous attempts to characterize the memory in ESN used an annealed approximation to simplify the problem. Our approach, however, gives an exact solution for the memory curve in a given ESN. Our analytical results agree very well with numerical simulations. However, discrepancies between the analytical memory curve and the fundamental limit of memory capacity hint at a hidden process that prevents the memory capacity from reaching its optimal value. We leave careful analysis of this deficiency for future work. In addition, we will study the presented framework to gain understanding of the effect of the structure of the system on the memory. We are currently extending this framework to inputs with non-uniform correlation structure. Other natural extensions are to calculate expected performance for an arbitrary output function and

analyze systems with finite dynamical range.

7.1 Acknowledgments

We thank Lance Williams, Guy Feldman, Massimo Stella, Sarah Marzen, Nix Bartten, and Rajesh Venkatachalapathy for stimulating discussions. This material is based upon work supported by the National Science Foundation under grants CDI-1028238 and CCF-1318833.

References

- [1] N. Bertschinger and T. Natschl ger. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7):1413–1436, 2004.
- [2] J. Boedecker, O. Obst, N. M. Mayer, and M. Asada. Initialization and self-organized optimization of recurrent neural network connectivity. *HFSP Journal*, 3(5):340–349, 2009.
- [3] L. B sing, B. Schrauwen, and R. Legenstein. Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. *Neural Computation*, 22(5):1272–1311, 2010.
- [4] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar. Information processing capacity of dynamical systems. *Sci. Rep.*, 2, 07 2012.
- [5] S. Ganguli, D. Huh, and H. Sompolinsky. Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences*, 105(48):18970–18975, 2008.
- [6] M. Hermans and B. Schrauwen. Recurrent kernel machines: Computing with infinite echo state networks. *Neural Computation*, 24(1):104–133, 2013/11/22 2011.
- [7] H. Jaeger. Short term memory in echo state networks. Technical Report GMD Report 152, GMD-Forschungszentrum Informationstechnik, 2002.
- [8] H. Jaeger. Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the “echo state network” approach. Technical Report GMD Report 159, German National Research Center for Information Technology, St. Augustin-Germany, 2002.
- [9] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.
- [10] W. Maass, T. Natschl ger, and H. Markram. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–60, 2002.
- [11] T. Natschlaeger and W. Maass. Information dynamics and emergent computation in recurrent circuits of spiking neurons. In S. Thrun, L. Saul, and B. Schoelkopf, editors, *Proc. of NIPS 2003, Advances in Neural Information Processing Systems*, volume 16, pages 1255–1262, Cambridge, 2004. MIT Press.
- [12] A. Rodan and P. Ti o. Minimum complexity echo state network. *Neural Networks, IEEE Transactions on*, 22:131–144, Jan. 2011.
- [13] H. O. Sill n, R. Aguilera, H. Shieh, A. V. Avizienis, M. Aono, A. Z. Stieg, and J. K. Gimzewski. A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing. *Nanotechnology*, 24(38):384004, 2013.
- [14] G. Snider. Computing with hysteretic resistor crossbars. *Appl. Phys. A*, 80:1165–1172, 2005.
- [15] G. S. Snider. Self-organized computation with unreliable, memristive nanodevices. *Nanotechnology*, 18(36):365202, 2007.
- [16] D. Snyder, A. Goudarzi, and C. Teuscher. Computational capabilities of random automata networks for reservoir computing. *Phys. Rev. E*, 87:042808, Apr 2013.
- [17] O. L. White, Daniel D. Lee, and H. Sompolinsky. Short-term memory in orthogonal neural networks. *Phys. Rev. Lett.*, 92:148102, Apr 2004.

- [18] I. B. Yildiz, H. Jaeger, and S. J. Kiebel. Re-visiting the echo state property. *Neural Networks*, 35:1–9, 2012.